Hindawi Publishing Corporation
Mobile Information Systems
Volume 2015, Article ID 463791, 11 pages
http://dx.doi.org/10.1155/2015/463791

Research Article

eMatch: An Android Application for Finding Friends in Your Location

Georgia Athanasopoulou and Polychronis Koutsakis

School of Electronic and Computer Engineering, Technical University of Crete, 73100 Crete, Greece

Correspondence should be addressed to Polychronis Koutsakis; polk@telecom.tuc.gr

Received 16 August 2015; Revised 7 November 2015; Accepted 25 November 2015

Academic Editor: Salvatore Carta

The work presented in this paper is twofold. We first outline the architectural design, the functional requirements, and the user interface of eMatch, an Android application which was inspired by the idea of fighting the loneliness we all witness in large cities. eMatch has the goal of connecting people with common interests that happen to be in the same geographical area. We then propose EgoSimilar, a new algorithm which computes the similarity between users and is implemented in eMatch. The algorithm is compared against two other well-known and widely used similarity computation methods and is shown to outperform them in terms of the most significant metrics used in our study.

1. Introduction

The majority of social media applications help people communicate, but only virtually, behind their computer screens. Another very common image in large cities, besides the one of people behind their screens, is that of people sitting alone in restaurants, coffee shops, and public transportation. Hence, although social networks offer the ability to socialize and form "virtual" relationships, in real world people often stay alone and distant from each other. Loneliness in our time has grown as bad as becoming a "social disease" [1].

The above thoughts and observations motivated us to design and develop an application, eMatch (electronic Match), the goal of which is to fill the gap between the virtual and the real world. Its purpose is to find people in the same geographical area, compare their interests, and suggest potential friends that the user can meet. This approach is based on the fact that friends tend to share common interests and activities, which has been shown in important research work on personality similarities and friendship dating back to the 70s [2, 3]. In the rest of the paper, when using the phrase "finding friends" we are referring to people with common interests, likes, and dislikes.

In eMatch, in order to compare people's interests, users rate a number of default interest categories, such as "Movies" and "Sports," while they can add and rate items to each one of them. For example, a user could rate the category "Sports" with "7" on a scale of 1 to 10 and add to this category the item "football" with rating "9." Based on this type of rating, the application's algorithm computes users' matching in order to suggest potential friends. Unlike other approaches in the literature, our algorithm, EgoSimilar, takes into account the popularity of the items that have been rated in its computations.

In the following section, we examine the related work in applications supporting tracking of potential friends based on common interests in the same geographical area. Section 3 analyzes the functional requirements of the application describing the use cases in detail as well as the basic principles of the user interface and the methods used in the design process. We describe the architecture of eMatch in Section 4 together with the tools used. In Section 5, our matching algorithm, EgoSimilar, is presented. In Section 6 we present and discuss the comparison of the results when using eMatch with EgoSimilar versus using it with other matching algorithms. Finally, Section 7 presents the conclusions of our work and our next steps in improving and evaluating eMatch.

2. Related Work

Although some applications related to eMatch have been developed (e.g., [4–6]) they focus more on tracking existing friends over a geographical map and less on finding and suggesting friends based on common interests, located in close geographical areas. The "Find my friends" application, in particular, which is developed by Apple, is currently installed automatically with iOS9. Facebook also added the use of a map for locating friends but finally deactivated it for security reasons [7, 8].

Two applications that are closer to eMatch in terms of their goals are the Gourmet Groups application [9] and the Youhoo application [10]. The Gourmet Groups application aims at helping people with similar interests get together at restaurants, lounges, and bars to enjoy food and drinks and is combined with a ticketing application to book seats in the specific venues and in significant events in the city. Therefore, the application does not use the current location of the user, but rather serves as a general application of connecting people with similar interests and of promoting specific venues. The Youhoo application is the closest to eMatch among all current applications in iOS and Android that are related to finding friends in an area near the user. Its goal is to create circles of people with common interests in an area. However, Youhoo profiles are created from Facebook; therefore, users who do not use Facebook are excluded from using the application, and users who wish to create a different profile or share only specific information in their Youhoo profile cannot do so. Additionally, the circles of people with common interests created by the application are quite generic or one-dimensional, for example, students in the same university, people working in the same field, and fans of a specific singer. On the contrary, eMatch computes the match between users based on the whole profile that the users wish to share through the application and of course allows users to create a profile that is independent from any other application.

The information location is used in eMatch only for practical reasons. In particular, eMatch uses the location information in order to locate potential friends in the same area (a 16 km radius is used as default for the city of Chania, Crete, Greece, where we initially tested our application) and not for tracking on the map and revealing the user's location as other applications do. The goal of eMatch is to facilitate potential friends to meet and introduce themselves to each other if they so wish. The user's location is considered private and sensitive information and is treated that way. In order to minimize security risks we have taken the two following measures: Firstly, we offer the choice of "Non-Visible" status under which the user's device will be invisible to the other devices even though he/she uses the application. This means that the "Non-Visible" option will hide the user from the application's matching results. Secondly, the user's interests and ratings are not visible to anyone except those users that he/she has designated as friends in the application. The only information that is public is the matching percentage for all pairs of "Visible" users inside the geographical area. In this way the individual's privacy is preserved.

3. Functional Requirements and User Interface

In this section we describe the most basic features of eMatch as well as the methods used during the user interface design procedure.

3.1. Functional Requirements. The system's functional requirements are the following:

(1) user registration;

(2) creation and administration of a personal profile: in order to introduce themselves to others, users create a personal profile. The profile contains two basic types of information:

   (a) personal information such as name and surname, professional or educational information, or a profile photo,

   (b) information about the user's interests: this is the fundamental type of information the users of eMatch should provide. The interests are divided into 9 basic categories: Movies, Music, Books, Games, Sports, Science, Shopping, Food, and Travel. The user rates each one of these categories based on how much he/she likes it, on a rating scale of 1 to 10, where 1 corresponds to hate, 5 to indifference, and 10 to love. Subsequently, for each one of the nine categories, users can insert items for which they have a strong opinion on (positive or negative) and rate them accordingly. In each category there are also default items inserted into the system, to ease the users' understanding of the application and to help them point out their preferences without having to type everything themselves. Users can choose to rate some or all of the default items or they can ignore them completely and rate only their own entries;

(3) creation and administration of the social network: every user creates a personal social network that includes the people that the user connects with. The network's nature is like the ones known from other social networks (Facebook, Twitter, etc.). A user connects with friends with a first degree relationship, with his friends of friends with a second degree and so on. A first degree relationship is of course formed only if both users agree.

(4) users' matching: this functionality comprises the ultimate purpose of our application which is the recommendation of potential friends based on a best-matching procedure of the application's users with respect to their geographical location. Matching is based on user ratings; therefore, the more data users decide to enter in their profile regarding their interests, hobbies, likes, and dislikes, the more accurate the matching results will be;

(5) searching users based on personal information (i.e., name, e-mail);

(6) communication with other users via personal messages;

(7) personal preferences configuration: a basic option that this functionality offers is the choice of the "Visible" and "Non-Visible" status we mentioned in Section 2 where a "Non-Visible" status means that the users can search for potential friends but their profile will be excluded from the results of other users' searches.

3.2. User Interface Design. Here, we describe the basic methods and design principles followed for designing the user interface. The main factor taken into consideration is that the application's target groups are smartphone users who generally do not have the same level of expertise and education. We designed the application considering the worst case scenario, that is, users inexperienced in technological aspects that have only elementary education. However, for the proper use of the application it is necessary that the user has basic knowledge of English.

Below are the basic rules followed when designing the user interface [11].

(1) *Template.* The strict definition of the template is essential for maintaining the rest of the rules and the basis of a good user interface. The content of structure and options is similar to that proposed to designers of Android Applications [12]. More specifically, emphasis was given to place all menu options on a bar at the top of the screen. The reason is that the mobile screen is small and it must contain as much useful information as possible.

As shown in Figure 1, the logo of eMatch is permanently placed on the left of this bar. When clicking on the logo, the action "back" is executed, in order to return to the previous screen. Placing the logo on the top left of the screen reminds users which application they are running. The choice of that side was made based on experiments which have shown that when users see a page, they start to read/look from the top left corner [11, 13]. Beside the logo, a text appears that serves as a "path" which helps users understand which functionality is being executed at every moment. For example, when a user sees the "personal profile" screen, then the text path is "My Profile." On the right corner of the bar an icon is placed that is used as a menu. When clicking on the icon, a pop-up window emerges which shows all the menu options of the user at the moment. For example, if the user is on the "personal profile" screen, then one of the options is "Edit Interest Categories." On the left of that menu icon, a second static icon is placed that serves to switch the user's status from "Visible" to "Non-Visible" and vice versa. The rest of the screen, below the bar, is the "Active Region" which is used to present nonstatic information.

(2) *Informing Users.* Informing the users about their conceptual position in the application is equivalent to providing answers to the following two simple questions, which all users have regardless of their experience: "Where am I" and "What exactly has happened." The answer to the question "Where am I" is given from the logo of the application and from the "path," which are placed permanently on the top of the screen and inform users about which application is used, as well as the kind of information presented in the current screen. The answer to the question "What exactly has happened" is given from informational messages that the system displays every time a user completes a specific action.

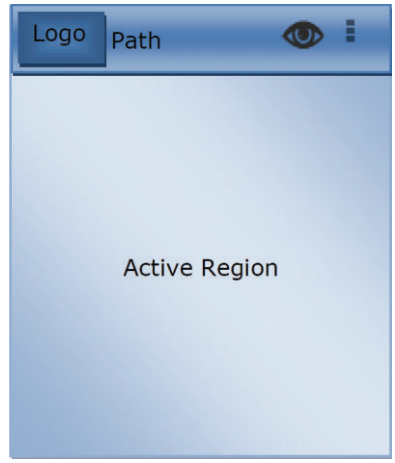


Figure 1: Template.

(3) *Choosing Colors.* The colors chosen for this particular application are in blue hues, because the blue color causes a sense of security, confidence, concentration, and independence to users [14, 15]. The background color is lilac, while the logo's color is electric blue in order to have a high contrast and be distinctive from the other items displayed on screen. The color of the text shown in the pages is black, with some exceptions where the color is gray or blue in order to distinguish them from the rest of the text (labels, matching results). In case of error the colors used are in red hues in order to create the sense of danger and error [8].

(4) *Consistency.* Consistency is very important when designing the user interface [16]. The entire application was designed following the pattern presented above. This helps users to avoid "getting lost" while exploring the application. So the information is presented in the same way across all the screens; that is, the same rules are followed when an error occurs, when helping the user, when presenting nonstatic information, and when displaying the menu.

The steps followed when designing the user interface are as follows.

(a) *Storyboards [17].* Storyboards are small predefined "stories," execution scenarios, as indicated by their

name. Basically each "story" corresponds to the steps that should be followed to complete one of the functional requirements of the application, from which we choose the most representatives ones. We made a rough design of the screens (each step/screen that leads to the next) on paper to clarify the order that should be followed in order to complete each story. The use of Storyboards and the evaluation of them are very important steps when designing a usable application, because usability problems can be found even during the simple design of the screens on paper.

(b) *Pluralistic Walkthrough [18, 19].* Pluralistic walkthrough is a methodology where real users evaluate Storyboards. This application is developed to be used by a wide range of users, so the users we selected for our evaluation (seven in total) had widely different characteristics (i.e., age, educational level). Additionally to the actual users, the evaluation involved one human specialist who is an expert in HCI (Human Computer Interaction).

(c) *Implementation.* When step (2) was completed, we made the necessary changes to the user interface design and moved on to its implementation using xml for Android. So, for each functional requirement we designed all the necessary xml files, since each one corresponds to a different graphical interface in the Smartphone.

4. Architectural Design and Tools

In order for an application such as eMatch to run, it is necessary to use a central server which stores all users' data and executes certain algorithms. The architecture of eMatch is divided into three layers as shown in Figure 2. Layer 1 is from the Smartphone side and Layers 2 and 3 are from the server side. Each layer communicates only with its adjacent layers.

First Layer. The first layer includes the interaction with users, where all the useful information is presented and received via the xml pages. Hence, any exchange of information with the user is done only through these pages, the information of which is managed from java classes, the Activities. The Activities are responsible for managing all data between the users and the application, and more specifically to present, edit, add, and read dynamic information via the user's screen (i.e., username and ratings). The first layer also includes the Services (also java classes) which represent services that are not interactive with the user and operate in the background. One example of such a service is the one which informs the server about the user's location at regular time intervals. The Activities and the Services communicate with the server through HTTP Requests and Responses. All information between server and mobile device is coded in accordance to the JSON standard [20].

Second Layer. The second layer of the architecture includes the Controller and the Event Handlers which constitute the main part of the application's business logic. Event Handlers are java classes, which may execute functional requirements or other useful operations of application. The Controller is a java class which handles the HTTP Requests and Responses and triggers the corresponding Event Handler, for the operation that should be executed.

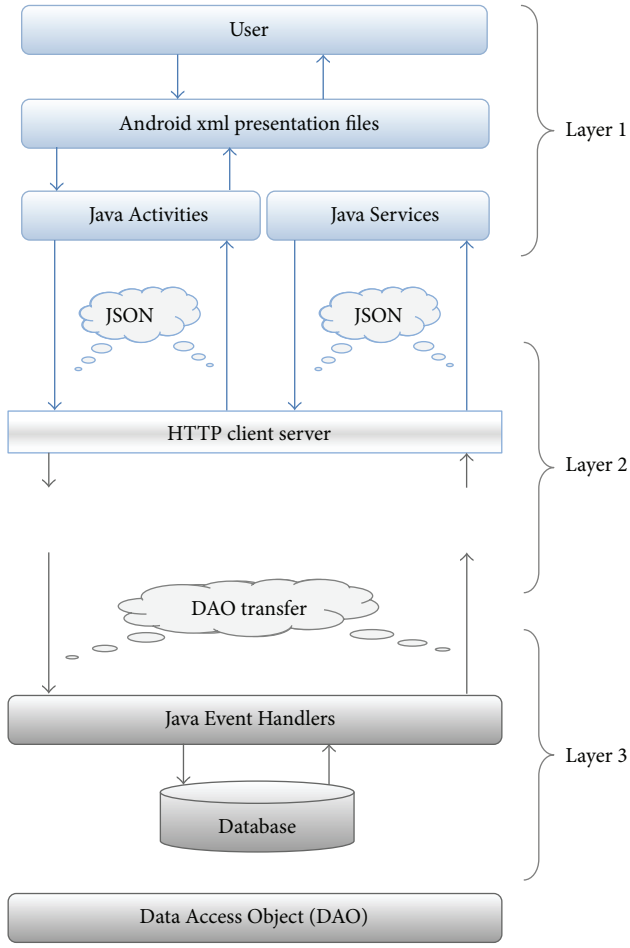


Figure 2: eMatch architecture.

Third Layer. The third layer of architecture includes the Data Access Objects (DAOs) and the database which stores all the necessary data of the application. DAOs are java classes used to communicate with the database; actually DAOs contain the appropriate functions for inserting, deleting, and editing the database data. Communication between the Event Handlers and DAOs is done through DAO Transfer, which are java classes that only transfer data between these two layers.

The advantage of using this layer is that it separates the business logic of the application and the database. Hence, if the database is changed then the only thing that should also change is the DAOs.

The tools used to develop this application are Eclipse IDE for implementing Android xml files, Java Activities, and Java Services; Netbeans IDE for implementing java code of Controller, Event Handlers, DAOs, and DAO transfer classes; and MySql database.

5. Matching

In this section we describe and discuss our "matching" algorithm, that is, our proposed algorithm for computing the similarity between users based on their interests and preferences. We also present, briefly, two other widely used methods for assessing similarity, that is, the Pearson Correlation and the Cosine Measure. All three approaches were implemented in eMatch, in order to find potential friends based on user ratings. These algorithms should run from the server side; their running at the smartphone would be uneconomic, and also battery and time consuming, since it would constantly require data transfers via mobile internet and many calculations to be executed.

For the matching algorithm to run at the server, the mobile must have internet access and at least one location provider activated. It should also store periodically (every 10 minutes is a reasonable interval) the geographical location of the user.

5.1. EgoSimilar. EgoSimilar is our proposed algorithm. It takes the following rationale into account:

(a) The matching is done in an "egocentric" way because each user should search friends based on his/her own criteria and interests. Thus, the matching percentage between two users that will appear on each user's screen will most likely be different. This makes sense in our view, given that in real life we operate in the same way, choosing the people around us based mainly on our own needs and interests, not on how compatible *we* are with their needs. Hence, if, for example, user X has one active category of interest while user Y has five, the matching percentage (X, Y) will be based on that one category, while the matching percentage (Y, X) will be based on all five, leading to different results showing on each user's screen. The two similarity measures against which EgoSimilar will be compared are also computed in the same way.

(b) More popular items (popular in the sense that they are rated positively or negatively by many users) should not affect matching results as much as less popular items do, if users "agree" on them. The reason is that even if users share, for example, a favorable opinion on very well-known band, book, movie, and so forth, this does not really give a substantial hint that their tastes match in general. A similar case regarding relatively unknown band/book/movie gives a much stronger indication of common interests.

(c) The rating choices of users are on a scale from 1 to 10. Consequently the maximum rating difference will be 9 and the weight of one unit in rating difference will be 1/9≈0.11. This weight should be included, in our view, in the computation of the similarity between users.

The steps followed by eMatch in computing the matching between users are described below. The first three steps are followed regardless of the matching computation method, which is implemented in step (4).

Let X be the user who runs the application; therefore, the matching is done according to X's tastes:

(1) Check if the user's location is stored. If not, inform the user; else go to the next step.

(2) Find users that are in close geographical proximity with user X.

(3) Find all the active interest categories of user X.

(4) The matching in EgoSimilar is computed as follows: for each user Y found in step (2), calculate the Matching(X, Y) as follows:

$$\frac{1}{k_X}\sum_{c=1}^{k_X}\left[w_1\left[1-0.11d_1\left(X,Y,c\right)\right] + \frac{w_2}{n_X^c}\sum_{i=1}^{n_X^c}\left[1-0.11d_2\left(X,Y,c,i\right)\right]\right], \tag{1}$$

where k_X is the number of active categories of user X, $k_X \in [1, 9]$. w_1 is the weight attributed to the general rating of a category. In our case, $w_1 = 0.25$, as we consider the "general" matching of users (e.g., both of them loving movies), to be of smaller importance, as their specific tastes in that category may differ significantly or even completely. Experiments with other values of w_1 are discussed in Section 6. w_2 is the weight of the ratings of all individual items of a category (in our case, $w_2 = 0.75$). n_X^c is the number of items user X has inserted in category c. $d_1(X, Y, c)$ is a function which computes the absolute difference in ratings between users X and Y for the cth activated category of user X. If user Y has deactivated the specific category, then we set $(1 - 0.11 \cdot d_1(X, Y, c))$ equal to zero. $d_2(X, Y, c, i)$ is associated with the ith item inserted by user X in cth activated category and denotes the distance of ratings between users X and Y for the specific item. We set $(1 - 0.11 \cdot d_2(X, Y, c, i))$ equal to zero if user Y has not rated this item; otherwise, $d_2(X, Y, c, i)$ is calculated, taking into account the popularity of the specific item, as follows:

(a) Initialize $d_2(X, Y, c, i)$ as the absolute difference in ratings between users X and Y for this item.

(b) Let m be the number of users that have inserted this item, and let n be the number of users that have inserted items in the cth activated category of user X. Then, the popularity weight of the specific item is defined as $W_i^c(X) = m/n$. An item is assumed to be popular if $W_i^c(X) > 0.5$, which means that more than half of the users that "voted" for this category have inserted the specific item (with either negative or positive rating).

(c) $d_2(X, Y, c, i)$ is adapted with respect to the popularity of the item and the rationale explained above as follows.

If $(W_i^c(X) > 0.5$ and $d_2(X,Y,c,i) < 5)$, then $d_2(X,Y,c,i) = d_2(X,Y,c,i) + W_{\text{change}} \cdot d_2(X,Y,c,i)$. This states that since this item is popular and the ratings of users are close, then this item should not affect matching results as much as less popular items do. Therefore, the distance of the ratings between users X and Y must be increased in order to decrease their matching. This increase is implemented via the W_{change} weight, the value of which we will discuss in Section 6.

If $(W_i^c(X) > 0.5$ and $d_2(X,Y,c,i) \geq 5)$, then $d_2(X,Y,c,i)$ does not change.

If $(W_i^c(X) \leq 0.5$ and $d_2(X,Y,c,i) < 5)$, then $d_2(X,Y,c,i) = d_2(X,Y,c,i) - W_{\text{change}} \cdot d_2(X,Y,c,i)$.

This indicates that since this item is not popular and the ratings of users are close, then this item should affect matching results more than the popular items do. Accordingly, the distance of the ratings between users X and Y must be decreased in order to increase their matching. This is implemented once again via the W_{change} weight.

If $(W_i^c(X) \leq 0.5$ and $d_2(X,Y,c,i) \geq 5)$, then $d_2(X,Y,c,i) = d_2(X,Y,c,i) + W_{\text{change}} \cdot d_2(X,Y,c,i)$. Similarly, in the case where the item is not popular and the ratings of users are not close, we infer that this is an indication of users that do not have common interests. So, by increasing the distance of their ratings, their matching is decreased.

The complexity of the algorithm is $O(pqr)$, where p is the number of the users, q is the number of categories (in our case, nine), and r is the maximum number of items inserted in one of the categories.

5.2. Pearson Correlation. The second approach for computing users' similarity, which was implemented and tested in eMatch, is the Pearson Correlation [21]. The Pearson Correlation does not take item popularity into account, as EgoSimilar does. A positive correlation shows similarities between users, while a negative value shows that these users are not similar and their interests are different. Below we describe how the Pearson Correlation is computed.

Let X and Y be two users of eMatch. The steps followed in order to compute their matching percentage are as follows:

(1) Find all the active interest categories of user X and all the items that X has inserted to each category and store the ratings of each element to a vector $\mathbf{x}$ (an element is either a category or an item; a separate vector is constructed for categories and another for items).

(2) Create a vector $\mathbf{y}$ that has equal size with vector $\mathbf{x}$. For each element of the previous step check if user Y has rated it. If yes, the corresponding position of vector $\mathbf{y}$ is assigned with Y's rating; otherwise, it is assigned a zero value.

(3) Let n be the length of vectors $\mathbf{x}$, $\mathbf{y}$, and denote by $\overline{x}$ the mean value of vector $\mathbf{x}$ and $\overline{y}$ the mean value of vector $\mathbf{y}$; then the Matching(X,Y) is computed as

$$\text{Matching}(X,Y) = \frac{\sum_{i=1}^{n} (\mathbf{x}[i] - \overline{x})(\mathbf{y}[i] - \overline{y})}{\sqrt{\sum_{i=1}^{n} (\mathbf{x}[i] - \overline{x})^2 \sum_{j=1}^{n} (\mathbf{y}[j] - \overline{y})^2}}. \quad (2)$$

In order to be able to make a meaningful comparison between EgoSimilar and the Pearson Correlation, we have computed the matching between categories and the matching between items separately with the use of the Pearson Correlation, using the same w_1 and w_2 weights (0.25, 0.75) as in the EgoSimilar implementation. The same weights were chosen when the Cosine Similarity (described below) was used as a similarity computation method.

5.3. Cosine Similarity. Another well-known similarity measure between two vectors is the Cosine Similarity [21]. The steps followed are exactly the same, as in the case of the Pearson Correlation, for the construction of vectors $\mathbf{x}$, $\mathbf{y}$. Then, the Matching(X,Y) is computed as follows:

$$\text{Matching}(X,Y) = \frac{\sum_{i=1}^{n} \mathbf{x}[i]\,\mathbf{y}[i]}{\sqrt{\sum_{i=1}^{n} \mathbf{x}[i]^2 \sum_{j=1}^{n} \mathbf{y}[j]^2}}. \quad (3)$$

A disadvantage of using Cosine Similarity in eMatch is that collinear vectors lead to 100% similarity. So, if $\mathbf{x}$'s ratings are 1 and $\mathbf{y}$'s 10, then the matching result is 100% which is a significant error, although in reality this scenario is rarely encountered.

5.4. Clustering. In order to examine the results of the above algorithms, users were separated into groups via the well-known K-means clustering algorithm [22], using the matching percentages derived by each of the three similarity computation approaches. Letting K be the number of clusters, the steps of the K-means algorithm are as follows:

(a) Create K initial centroids, one for each cluster. Initialize the ratings of each centroid for all categories and for all distinct items inserted in each category by the users.

(b) Assign each user to the cluster with the closest centroid. The distance of a user and a centroid was computed, using one of the three algorithms described above, as Matching(X,Y), where X is the real user and Y is the centroid.

(c) When all users have been assigned to a cluster, recalculate the ratings of the K centroids, by taking for each element (item or category) the average rating of real users' in the group.

(d) Repeat steps (3) and (4) until the centroids' ratings no longer change.

Table 1: Overall matching results.

	Matching metrics		
	AM	AMC	AMnC
EgoSimilar ($W_{change} = 0.1$)	48.8%	56.0%	48.5%
EgoSimilar ($W_{change} = 0.2$)	48.1%	55.3%	47.8%
EgoSimilar ($W_{change} = 0.3$)	47.5%	54.5%	47.2%
Pearson	8.3	11.9	8.2
Cosine	68.3%	76.5%	68.0%

The procedure will always terminate, but K-means does not necessarily find the optimal configuration. A disadvantage of K-means is its sensitivity to the random initialization of cluster centroids; generally initial centroids should be "far apart." We addressed this issue in our evaluation by using different centroids and computing average results over 10 independent runs. Other clustering algorithms can also be used, but their comparison with K-means is out of the scope of this paper.

6. Evaluation

For the evaluation of the matching algorithms we collected data from 57 users (ages 18–40) who are friends in real life. The collected information consisted of the Activation/Deactivation of the 9 interest categories, the Ratings for all active categories, and the Ratings for the individual items in all the active categories. The items rated in each category were either new insertions by the users or as many of the default items as the users wished to rate. We collected data from 57 users.

The reason we chose to collect data mainly from groups of friends (still, some of the participants had no connection to other users) was that in this way it would be feasible to

(a) evaluate whether the similarity computation methods would be able to "discover," through higher matching values, existing friendships

(b) reveal which users that were currently unknown to each other could potentially become friends.

In total, 43 of the 57 participants had at least one connection. These 43 participants formed in total 64 first level connections. The statistics of the ratings of all 57 users were mean rating value 6.6 and standard deviation 2.7. These statistics imply that users mainly rate items that they like, instead of taking the time to state what their dislikes are as well. To compare the results we ran the K-means clustering algorithm, each time with a different similarity computation approach (EgoSimilar, Pearson, and Cosine). We derived results for a number of clusters K equal to 3, 5, 7, and 10, respectively, in order to evaluate how (and if) the number of clusters influences the user matching.

In Tables 1–7, where our results are presented, the columns contain the metrics used in our study, which are the following:

(a) id: cluster id where id $\in [1, K]$,

Table 2: Average friends' placement.

	Placement
EgoSimilar ($W_{change} = 0.1$)	20.12
EgoSimilar ($W_{change} = 0.2$)	20.04
EgoSimilar ($W_{change} = 0.3$)	19.91
Pearson	21.40
Cosine	21.77

Table 3: EgoSimilar ($W_{change} = 0.1$).

id	Matching metrics per cluster						
	$N1$	$N2$	$N3$	AVC (%)	AM (%)	AMC (%)	AMnC (%)
1	14	12	4	30.4	50.7	55.9	50.5
2	13	8	4	18.5	50.3	63.4	49.6
3	4	3	0	0.0	25.1	—	25.1
4	22	18	8	41.7	57.6	58.3	57.6
5	4	2	0	0.0	41.9	—	41.9
Σ	**57**	**43**	**16**				

Table 4: EgoSimilar ($W_{change} = 0.2$).

id	Matching metrics per cluster						
	$N1$	$N2$	$N3$	AVC (%)	AM (%)	AMC (%)	AMnC (%)
1	8	7	1	9.5	30.7	33.4	30.6
2	19	14	5	32.7	53.4	55.0	53.4
3	0	0	0	—	—	—	—
4	24	17	10	36.2	52.5	64.3	52.0
5	6	5	0	0.0	56.0	—	56.0
Σ	**57**	**43**	**16**				

Table 5: EgoSimilar ($W_{change} = 0.3$).

id	Matching metrics per cluster						
	$N1$	$N2$	$N3$	AVC (%)	AM (%)	AMC (%)	AMnC (%)
1	9	8	2	16.4	47.7	53.1	47.5
2	2	0	—	—	27.7	—	27.7
3	34	26	25	53.5	54.7	62.2	54.3
4	6	5	1	13.3	45.9	61.7	44.8
5	6	4	0	0.0	29.3	—	29.3
Σ	**57**	**43**	**28**				

(b) $N1$: the number of users in the cluster,

(c) $N2$: the number of users in the cluster that have a network (i.e., they are connected with at least one other user, who may be in that cluster or in another one),

(d) $N3$: the number of users in the cluster that are connected in reality as friends,

(e) AVC (Average Valid Connections): for each user in a cluster we computed the percentage of their connections that are included in the specific cluster and derived the average percentage,

Table 6: Pearson.

id	Matching metrics per cluster						
	$N1$	$N2$	$N3$	AVC (%)	AM	AMC	AMnC
1	22	16	6	27.8	10.1	25.5	9.7
2	6	5	1	12.9	21.3	24.7	21.1
3	13	10	3	16.5	14.6	14.6	14.6
4	6	4	0	0.0	17.0	—	17.0
5	10	8	3	37.5	10.8	7.9	11.0
Σ	**57**	**43**	**13**				

Table 7: Cosine.

id	Matching metrics per cluster						
	$N1$	$N2$	$N3$	AVC (%)	AM (%)	AMC (%)	AMnC (%)
1	3	2	0	0.0	47.1	—	47.1
2	25	18	15	51.0	70.5	74.6	70.3
3	4	3	0	0.0	61.7	—	61.7
4	12	11	6	36.4	83.8	87.2	83.5
5	13	9	1	3.7	57.9	86.6	57.5
Σ	**57**	**43**	**22**				

(f) AM (average matching): this is the average matching percentage of all users of the specific cluster,

(g) AMC (Average Matching of Connected users): this is the average matching percentage of all the connected users of the cluster,

(h) AMnC (Average Matching of not Connected users): this is the average matching percentage of all the users of the cluster who are not connected.

Considering the aforementioned mean and standard deviation of the users' ratings, we initialized the centroids' ratings randomly in the interval from 4 to 10. The Pearson Correlation and Cosine Similarity values range in the interval $[-1, 1]$ (multiplied by 100 to be able to make comparisons with EgoSimilar). However, matching values for the Cosine Similarity are bounded in the interval $[0, 100]$ because all the rating values are positive. The matching values for EgoSimilar are also bounded in the same interval. Table 1 presents the average matching results between each user and all other users, before any clustering is implemented.

The following initial conclusions can be derived by studying the above results:

(1) In all five implementations the matching is higher between connected users (i.e., users who have already formed relationships with each other) than between not connected users. This result is expected and intuitively simple to understand.

(2) The Pearson Correlation achieves the worst results in terms of distinguishing between connected and not connected users, via the matching computation. The difference between AMC and AMnC, with the use of EgoSimilar, is around 7.5% regardless of the value of W_{change}, and with the use of the Cosine Similarity it is 8.5%. When the Pearson Correlation is used this difference is 3.7 in the $[-100, 100]$ scale; therefore, it would be 1.85% if the scales were normalized to $[0, 100]$. Therefore, the Pearson Correlation fails to "discover" which users are already connected, a fact that according to the psychology literature should be evident from their preferences and hence their matching.

(3) EgoSimilar leads to equal AMC matching results with the Pearson Correlation, for $W_{\text{change}} = 0.1$ and to the smallest matching results for higher values of W_{change}. EgoSimilar also leads to the smallest average matching results for not connected users, for all values of W_{change}. Additionally, the average matching for both connected and not connected users decreases when W_{change} increases. The reason for these results is that most users that took part in the evaluation decided to rate most of the default items of the application. This made the default items popular, and popular items, in EgoSimilar, can either lead to a decrease in matching (if users have significantly different opinions on them) or to no change at all. The decrease is obviously larger for larger W_{change} values. This explains the apparent "strictness" of EgoSimilar, in comparison to the Pearson Correlation and the Cosine Measures (which however carries the problem of the collinear vectors, explained in Section 5). Based on the observed behavior of independent users who have downloaded the initial version of eMatch that is already available in Google Play [23], we expect that this strictness will be mitigated, as many users prefer to enter their own items instead of rating the default ones.

It should be emphasized, however, that the actual matching percentage is of little value. The only substantial effect that it might have, especially in the case of not connected users, is that a quantitatively higher percentage might be more intriguing for a user in order to decide to communicate with another user. What is really substantial is the order in which "matching users" appear on the user's screen, in decreasing percentages (high to low). A user X would obviously consider first the users with whom he/she has the highest matching, regardless of the actual matching percentage (unless the matching percentage is very low even for the "top matched" user, which would be discouraging). Also, in this matching list we would expect existing friends to place high. Hence, to evaluate the three similarity computation methods, we located in each user's matching list the position in which their existing friends were placed. The results are presented in Table 2, and they show that the best placement of existing friends is clearly achieved by EgoSimilar. The fact that the Pearson Correlation again exhibits worse performance than EgoSimilar can be explained if we consider that it evaluates how different the ratings of specific categories/item are, in comparison to the ratings that the users give on average. This makes it vulnerable when a few or even one extreme value is included in the ratings that are being compared; hence, for users with "outliers" in their ratings its results are

not consistent. On the other hand, both the Cosine Measure and EgoSimilar only examine the current ratings of each category/item, by each of the two users. EgoSimilar, however, tries to be more sophisticated by using weights based on the popularity of the rated items and on the rating scale. For this reason, the Cosine Similarity is shown to perform comparably to the Pearson Similarity and worse than all the implementations of EgoSimilar.

Tables 3–7 present our results when using K-means clustering with $K = 5$. Each row corresponds to one cluster/group of users. Firstly, by comparing the sum of the $N3$ columns, it is clear that the EgoSimilar algorithm outperforms the Pearson Correlation approach regardless of the W_{change} value; that is, it manages to include in the same cluster more people who are already connected as friends prior to the use of eMatch. Also, the results presented in Tables 3–7 show that EgoSimilar clearly outperforms the Cosine Measure as well, in terms of $N3$, when W_{change} is equal to 0.3 (28 versus 22 users). We have ran our experiments repeatedly, for various random initial centroids, and with 3 different values of W_{change}: $\{0.1, 0.2, 0.3\}$. In each case there was at least one value of W_{change} which achieved much higher $N3$ results than the Cosine Measure. The reason that the value of W_{change} achieving the best results was not the same in all the cases is the aforementioned sensitivity of the K-means clustering algorithm to the random initialization of the cluster centroids. If the centroids are adequately "far apart" then even a small value of W_{change} is enough to take advantage of the few nonpopular items, for which connected users share an interest, and to place the connected users in the same cluster. If, however, the centroids are not adequately "far apart" and the users' ratings contain outliers, then in order to enforce the clustering of similar users a larger value of W_{change} is needed. In order to overcome this problem, an intuitively simple solution would be to use default initial centroids, adequately spaced. This solution, however, is not practical because the choice of the default centroids would have to be made based on the dataset, which is not static as was the case with the 57 participants' data; it changes in real-time for the actual users of eMatch. Hence, this approach would require constant recalculations at the server. Therefore, given the results presented in Table 2, where EgoSimilar outperforms the other approaches regardless of the W_{change} value, and assuming that the more users download and run eMatch the more disparate the ratings will be, we use $W_{\text{change}} = 0.3$.

Finally, the comparison of EgoSimilar with the two other similarity computation methods in terms of the AVC, AM, AMC, and AMnC metrics shows that EgoSimilar is comparable with the Pearson Correlation for all values of W_{change} while it derives smaller matching percentages over all metrics when compared with the Cosine Measure. The Pearson Correlation once again fails to "discover" which users are already connected in two of the 5 five clusters (AMC $\leq$ AMnC), as shown in Table 6.

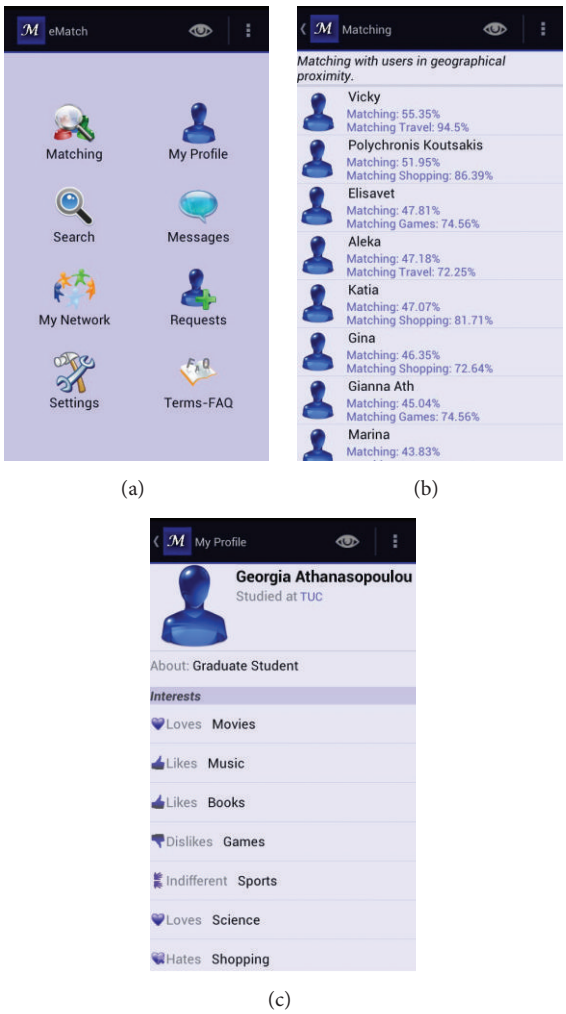
In Section 5 we mentioned that we used different values of w_1 and w_2 in our implementation. As expected, when using lower values for w_1 (hence, larger for w_2 since $w_1 + w_2 = 1$) the matching percentages decrease. The reason is that the matching of users is easier in the categories' field than it is in the items' field. Therefore, a smaller weight in the categories' matching leads to a lower overall matching, and a higher w_1 leads to higher overall matching, respectively. Indicatively, for $\{w_1 = 0.15,\ w_2 = 0.85\}$ the overall matching decreases by 3-4%, for all similarity computation methods.

Also, as mentioned earlier in this section, we derived results for a number of clusters K equal to 3, 5, 7, and 10, respectively, in order to evaluate how (and if) the number of clusters influences the user matching. Our results have shown that there is no qualitative differentiation in the matching results by changing the value of K. The only difference is quantitative: for $K = 7$ and $K = 10$ all approaches are shown to lead to smaller matching results than those achieved for $K = 5$, due to the fact that 1-2 very large clusters and a few very small or even empty ones were created. This is related, of course, to the fact that the number of users in our study (57) is not very large; therefore, a high number of clusters is useless. The results for $K = 3$ present no substantial differences with those for $K = 5$.

What is more important, however, is that EgoSimilar continues to outperform for all values of K both the Pearson Correlation and the Cosine Measure in terms of the $N3$ metric. The Cosine Measure continues to provide the highest matching percentages within its clusters, both for connected and for nonconnected users, for all values of K, while the respective results of the Pearson Correlation (normalized in the scale [0, 100]) are comparable to the EgoSimilar ones.

We should also mention that various approaches have been used in information retrieval and collaborative filtering which are relevant to our proposed algorithm. Clustering techniques have long been used in information retrieval to improve the performance of search engines [24]. However, the use of clustering techniques (balanced clustering, single link clustering, and group average clustering) to improve the performance of people matching has been associated with hierarchical agglomerative algorithms. Hierarchical clustering, however, is well known to underperform in comparison to K-means in the case of large datasets, due to its significantly increased execution time. Hence, in the case of a large number of users simultaneously running eMatch, K-means is the better choice.

The authors in [25] use ranking functions to propose a method that represents people's preferences in a metric space, where it is possible to define a kernel based similarity function; they then use clustering to discover significant groups with homogeneous states. The authors point out the success of the Pearson Correlation and the Cosine Similarity in order to make comparisons between the rating vectors of different users; they also use Cosine Similarity in their work. Still, their proposed class separation technique which utilizes Support Vector Machines (SVMs) becomes computationally complex and leads them to avoid using K-means, to decrease the computational complexity of the combination of K-means with their technique. Instead, they use a nonparametric pairwise algorithm, which yields a bipartition of the dataset into two clusters and then recursively proceeds to apply the partition mechanism to each of the resulting clusters. In order to ensure that only meaningful splits take place, the authors utilize a cross validation method from [26] which measures



(a) (b)

(c)

Figure 3: eMatch screenshots.

an index that can be read as a significance level; they then accept only splits whose level is above a threshold value. It is clear from the above that this method once again is much more computationally complex then the algorithm running on eMatch.

Figure 3 presents three screenshots from the use of eMatch: (a) the initial screen, (b) the matching percentages of a user X with users who are geographically close (overall matching plus the "best match category" between X and the other user), and (c) user X's profile.

7. Conclusions and Future Work

We have designed and developed a new mobile application, eMatch, which has the goal of helping users to find people with similar interests who are in geographical proximity, while respecting the users' privacy. The idea for eMatch was created by the fact that the contemporary way of life leads a large number of people to spend much time away from home, often alone among strangers. Therefore, it makes sense for them to connect right on the spot with someone close by who shares their interests. This is a decision that can be made quickly with the help of an intelligent application, as opposed to decisions regarding finding possible life partners, which would usually require much more thought and study from the user (other applications focus on this area). Even at home, however, users spend a large amount of time using their mobile devices. Therefore, even users who want to take their time with evaluating possible friends will have the opportunity to do so.

Besides providing the details on the architectural design, the functional requirements, and the user interface of eMatch, we have proposed a new algorithm, EgoSimilar, which computes the matching between users. Our algorithm clearly outperforms two of the most well-known similarity measures, the Pearson Correlation and the Cosine Measure, in regard to the most significant metrics used in our study. The 57 users who volunteered for the evaluation of the algorithm have been carefully monitored, stating their preferences and pointing out their existing friendships; we would like to thank them for their valuable help in the evaluation of the EgoSimilar algorithm.

The present work describes the first step in the design and evaluation of eMatch and our EgoSimilar algorithm. We intend to expand the evaluation by using a much larger pool of users. This is a time-consuming task, given that we need to find a large number of users with existing friendships who will be willing to use the application under our supervision, so that the experiment is conducted without errors. This work is currently under way.

In the second version of eMatch on Google Play, which is currently being prepared, we will allow users to choose between "seeing" only the users who belong in the same cluster as them or all users, that is, even those with whom they may have little or no common interests. The next step of our work will be the incorporation of semantic similarity computation algorithms into eMatch, to further improve the clustering and the implicit (via the matching percentage) friendship recommendations. The use of such algorithms is important, so that relevant concepts, names, and items will be linked automatically by the application (e.g., soccer and football, or soccer and Manchester United). Additionally, we plan on conducting a longitudinal study to examine when and how people find matches in mobile situations.

We also believe that eMatch can evolve so that, additionally to serving as a friend recommender system, it will function as an item recommender system as well, for items that are highly recommended by users who share a high matching percentage in specific categories.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] J. T. Cacioppo and W. Patrick, *Loneliness: Human Nature and the Need for Social Connection*, W. W. Norton & Company, 2009.

[2] S. W. Duck and G. Craig, "Personality similarity and the development of friendship: a longitudinal study," *British Journal of Social and Clinical Psychology*, vol. 17, no. 3, pp. 237–242, 1978.

[3] C. Werner and P. Parmelee, "Similarity of activity preferences among friends: those who play together stay together," *Social Psychology Quarterly*, vol. 42, no. 1, pp. 62–66, 1979.

[4] Parashar, "Nearby friend finder," https://play.google.com/store/apps/details?id=com.fsp.android.friendlocator.

[5] Sonar Media, Inc., "Sonar: Friends nearby", https://play.google.com/store/apps/details?id=com.valez.near.

[6] https://itunes.apple.com/us/app/find-my-friends/id466122094.

[7] http://www.wired.com/gadgetlab/2012/06/facebook-quietly-releases-find-friends-nearby-then-quietly-pulls-it/.

[8] http://www.dailymail.co.uk/sciencetech/article-2164366/Facebook-kills-Find-Friends-Nearby-feature-stalking-fears-GPS-app.html.

[9] http://appcrawlr.com/android/gourmet-groups.

[10] http://appcrawlr.com/android/youhoo.

[11] W. O. Galitz, *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*, Wiley, 2007.

[12] R. Meier, *Professional Android 4 Application Development*, John Wiley & Sons, New York, NY, USA, 2012.

[13] C. S. Campbell and P. P. Maglio, "Facilitating navigation in information spaces: road-signs on the World Wide Web," *International Journal of Human-Computer Studies*, vol. 50, no. 4, pp. 309–327, 1999.

[14] P. Russo and S. Boor, "How fluent is your interface?: designing for international users," in *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, pp. 342–347, Amsterdam, The Netherlands, April 1993.

[15] T. Soen, T. Shimada, and M. Akita, "Objective evaluation of color design," *Color Research & Application*, vol. 12, no. 4, pp. 187–195, 1987.

[16] C. M. Brown, *Human-Computer Interaction Design Guidelines*, Intellect Books, Bristol, UK, 1998.

[17] K. N. Truong, G. R. Hayes, and G. D. Abowd, "Storyboarding: an empirical determination of best practices and effective guidelines," in *Proceedings of the 6th ACM Conference on Designing Interactive Systems (DIS '06)*, pp. 12–21, ACM, June 2006.

[18] J. Nielsen, "Usability inspection methods," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '94)*, pp. 413–414, 1994.

[19] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pp. 249–256, ACM, 1990.

[20] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, "Comparison of JSON and XML data interchange formats: a case study," in *Proceedings of the 22nd International Conference on Computer Applications in Industry and Engineering (CAINE '09)*, pp. 157–162, San Francisco, Calif, USA, November 2009.

[21] L. Mekouar, Y. Iraqi, and R. Boutaba, "An analysis of peer similarity for recommendations in P2P systems," *Multimedia Tools and Applications*, vol. 60, no. 2, pp. 277–303, 2012.

[22] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained k-means clustering with background knowledge," in *Proceedings of the 18th International Conference on Machine Learning (ICML '01)*, vol. 1, pp. 577–584, 2001.

[23] https://play.google.com/store/apps/details?id=com.tuc.eMatch&hl=en.

[24] M. D. Dunlop, "Development and evaluation of clustering techniques for finding people," in *Proceedings of the 3rd International Conference on Practical Aspects of Knowledge Management (PAKM '00)*, Basel, Switzerland, October 2000.

[25] J. Díez, J. J. del Coz, O. Luaces, and A. Bahamonde, "Clustering people according to their preference criteria," *Expert Systems with Applications*, vol. 34, no. 2, pp. 1274–1284, 2008.

[26] S. Dubnov, R. El-Yaniv, Y. Gdalyahu, E. Schneidman, N. Tishby, and G. Yona, "A new nonparametric pairwise clustering algorithm based on iterative estimation of distance profiles," *Machine Learning*, vol. 47, no. 1, pp. 35–61, 2002.